Reconsidering Multi-Dimensional Schemas

Tim Martyn
Rensselaer at Hartford
martyn@rh.edu

Abstract

This paper challenges the currently popular "Data Warehouse is a Special Animal" philosophy and advocates that practitioners adopt a more conservative "Data Warehouse = Database" philosophy. The primary focus is the relevancy of Multi-Dimensional logical schemas. After enumerating the advantages of such schemas, a number of caveats to the presumed advantages are identified. The paper concludes with guidelines and commentary on implications for data warehouse design methodologies.

1. Introduction

Within the past decade, many scholarly and trade publications have advocated the position that data warehouse systems, designed to facilitate Decision Support Systems in general, and On-Line Analytic Processing (OLAP) in particular, are "special animals" that require special design methodologies to build special logical schemas.

"OLTP is profoundly different from dimensional data warehousing. The users are different, the data content is different, the hardware is different, the software is different, the management of the systems is different, and the daily rhythms are different." Kimball [5]

"Designing a data warehouse requires techniques completely different from those adopted for operational systems."
Golfarelli & Rizzi [2]

The primary purpose of this paper is to invite practitioners to reconsider this predominant "Data Warehouse = Special Animal" philosophy. Instead, practitioners will be encouraged to adopt a more fundamental, but apparently less popular, *"Data Warehouse = Database (DW=DB)"* philosophy. The essence of this philosophy is captured in the following quotes.

"Data warehouses and data marts are nothing more or less than SQL database systems."
Hellerstein, Stonebraker, and Caccia [3]

"The rules of logical design do not depend on the intended use of the database - the same rules apply, regardless of the kinds of applications intended. In particular, therefore, it should make no difference whether those applications are operational (OLTP) or decision support applications. Either way, the same design procedure should be followed." C. J. Date [1]

This paper will argue the merits of the DW=DB philosophy within the context of logical database design for implementation using a conventional relational DBMS. In particular, it will present a number of caveats relevant to the *presumed* advantages of Multi-Dimensional (MD) logical schemas that have found great application within the data warehouse environment.

Section 2 provides a brief overview of three logical schemas applicable within a data warehouse environment. Section 3 presents design criteria for evaluating the schemas. These criteria are applied in Section 4 to present arguments in favor of MD schemas and in Section 5 to present arguments against MD schemas. Section 6 considers the impact of query optimization on the design of logical schemas. Section 7 concludes with pragmatic guidelines and a few comments on design methodology.

2. Logical Schemas for Data Warehouse Applications

Two specialized logical database schemas have been proposed for data warehouse applications. These are the Star Schema and the Snowflake Schema. The more generic term "Multi Dimensional (MD) Schema" is used to collectively refer to both schemas. Third Normal Form (3NF) Schemas are also considered, even though many authors contend that such schemas are not appropriate for data warehouse applications. Figure 1 illustrates each type schema. Most data warehouse applications also include some pre-computed summary tables (materialized views), but we do not consider such tables to be part of the core logical schema.

Star Schema: Figure 1.A illustrates that a star schema has a large "fact" table in the center, with multiple "dimension" tables surrounding the fact table. There is a one-to-many relationship between each dimension table and the fact table. The fact table usually represents business transactions or events, or a snapshot summary of the transactions/events. Because most real world applications do not directly conform to a star structure, some dimension tables may not be in third normal form.

Snowflake Schema: Figure 1.B illustrates a snowflake schema that may be interpreted as an extension to a star schema. This figure illustrates that a star is situated in the center of the snowflake. The major extension is the presence of "outer-level" dimension tables. A path of one-to-many relationships from each outermost table to the central fact table represents a dimensional hierarchy. The presence of outer-level dimension tables usually reduces, but does not necessarily eliminate, the number of de-normalized tables.

3NF Schema: The term "3NF schema" refers to a logical schema where (almost) all base-tables are (at least) in third normal form. Figure 1.C illustrates a 3NF schema that is usually derived from a semantic data model such as an ER or UML Model. Note that Figure 1.C illustrates an explicit relationship between dimension tables (DEPT and REGION). Such relationships are not explicitly represented within MD schemas.

The schemas shown in Figure 1 are not semantically equivalent. The 3NF schema contains more information than the snowflake schema, which in turn contains more information than the star schema. However, in most cases, it is possible to represent the semantics of any application within any type of schema. For example, Figure 2 illustrates a star schema with de-normalized tables that represents the same information embodied within the 3NF schema shown in Figure 1.C.

Different kinds of schemas might be used for the different kinds of data stores found within a data warehouse environment. For example, it might be reasonable to utilize (i) a star schema in a data mart, (ii) a snowflake schema in a data warehouse and (iii) a 3NF schema in an operational data store. We emphasize that our analysis of logical database schemas is relevant regardless of the particular data store under consideration.

Figure 1A: Star Schema

PURCHASE (S#, P#, T#, PRICE) [fact]
TASK (T#, TNAME) [dimension]
PART (P#, PNAME, PWT) [dimension]
SUPPLIER (S#, SNAME, SZIP) [dimension]

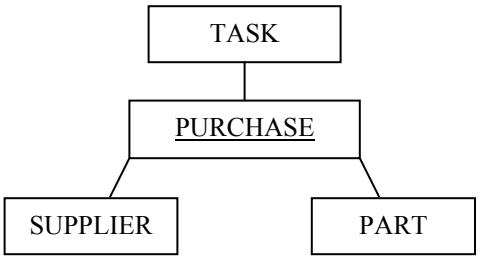

Figure 1B: Snowflake Schema

PURCHASE (S#, P#, T#, PRICE)
DEPT (D#, DNAME, DBUDGET)
PROJ (PJ#, PJNAME)
TASK (T#, TNAME, **D#, PJ#)**
PART (P#, PNAME, PWT)
REGION (R#, RNAME)
SUPPLIER (S#, SNAME, SZIP, **R#**)

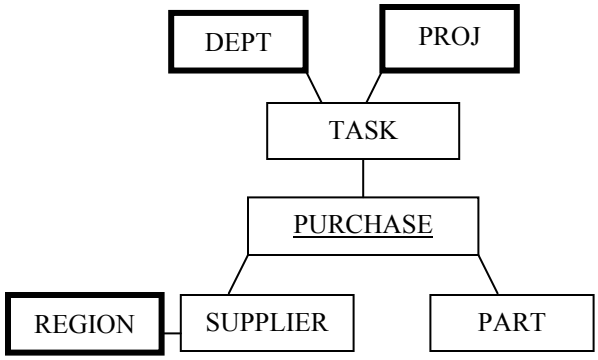

Figure 1C: 3NF Schema

Modify design for Figure 1.B. Include R# in DEPT.
DEPT (D#, DNAME, DBUDGET, **R#**)

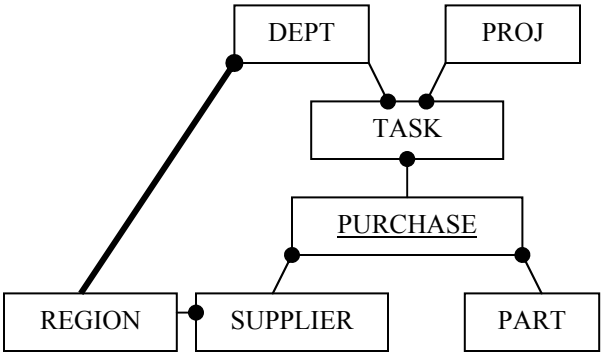

Figure 1: Three Possible Data Warehouse Schemas

3. Design Criteria

Any attempt to determine the "best" logical database schema must be based on some criteria. Three generic criteria, applicable to all information systems, including data warehouse systems, are proposed. The ideal system should be (i) correct, (ii) fast, and (iii) friendly.

Correctness Criterion: Correctness is the most important criterion because a friendly system that efficiently generates incorrect information is a failure. We include the notion of completeness within correctness.

Efficiency Criterion: A database warehouse is usually a very large database. Many queries will access large amounts of data, and usually involve multiple join operations. Hence, machine efficiency becomes a major consideration.

Usability Criterion: Within a DSS, users frequently formulate their own queries. Hence, usability becomes a major consideration.

Advocates of MD schemas often reference the "fast and friendly" criteria when they deprecate 3NF schemas that are directly derived from semantic data models. Kimball is most explicit on this matter.

"*The normalized structures must be off-limits to user queries because they defeat understand-ability and performance."* [6]

Kimball would not accept an incorrect design. His comments imply that, because the "fast and friendly" criteria are so important, a correct MD schema is preferred over a correct 3NF schema. The following section elaborates on the rationale for this position. The subsequent section identifies some important caveats. The primary objective of this paper is to invite practitioners to reconsider MD logical schemas by presenting both sides of the "3NF versus MD Logical Schema" debate. In simple terms, this debate can be summarized by two questions. Assuming that your design methodology has (somehow) produced a correct 3NF logical schema:

(i) Should you transform your 3NF schema into a MD schema to promote machine efficiency?

(ii) Should you transform your 3NF schema into a MD schema to promote human usability?

Figure 2: De-Normalized Star Schema

PURCHASE (S#, P#, T#, PRICE)
PART (P#, PNAME, PWT)
SUPPLIER (S#, SNAME, SZIP, R#, RNAME)
TASK (T#, TNAME, PJ#, PJNAME, D#, DNAME, DBUDGET, R#, RNAME)

4. Arguments for MD Schemas

MD schemas can be defended with respect to all three criteria. However, advocates of MD schemas emphasize efficiency and usability.

4.1 Correctness

Correct semantics can be represented with a MD schema even though some tables may not be in third normal form. This is acceptable because a data warehouse is generally a "read-only" database. Potential update anomalies can be addressed during the extract-transform-load operations that populate the data warehouse.

4.2 Efficiency

Given the very large size of many data warehouses, and the complexity of data warehouse queries, many designers implement MD schemas for performance reasons. Within this context, we consider each schema.

3NF Schema: A 3NF design is considered to be the least efficient design given the potential for a large number of join operations. Furthermore, some optimizers "stress out" when they encounter a query with many join operations and generate an inefficient query plan.

Star Schema: A star schema is generally considered to be the most efficient design for two reasons. First, a design with de-normalized tables encounters fewer join operations. Second, most optimizers are smart enough to recognize a star schema and generate access plans that use efficient "star join" operations. Kimball notes that a "standard template" data warehouse query directly maps to a star schema. [5].

Snowflake Schema: Sometimes a pure star schema might suffer performance problems. This can occur when a de-normalized dimension table becomes very large and penalizes the star join operation. Conversely, sometimes a small outer-

level dimension table does not incur a significant join cost because it can be permanently stored in a memory buffer. Furthermore, because a star structure exists at the center of a snowflake, an efficient star join can be used to satisfy part of a query. Finally, some queries will not access data from outer-level dimension tables. These queries effectively execute against a star schema that contains smaller dimension tables. Therefore, under some circumstances, a snowflake schema is more efficient than a star schema.

4.3 Usability

Non-technical users frequently formulate their own ad hoc queries against a data warehouse or data mart. This situation may justify MD schemas based on usability considerations. We consider each schema from a usability perspective.

3NF Schema: Figure 1.C clearly indicates that a 3NF schema is the most "complex" structure in the sense that it has the greatest number of rectangles and lines. Kimball's experience indicates that users cannot understand such complex designs.

"Normalized models, however, are too complicated for data warehouse queries. Users can't understand, navigate, or remember normalized models that resemble the Los Angeles freeway system." [6]

Star Schema: A number of usability advantages have been identified for star schemas.

- The star schema is the simplest structure in the sense that it has the smallest number of rectangles and lines.
- Because a star schema has the fewest tables, users execute fewer join operations. This makes it easier to formulate queries.
- The typical data warehouse query gracefully maps to the star schema.
- The star schema could serve as a generic logical schema for all data warehouses. If users become comfortable with this schema, their learning time for other star schemas would be reduced.
- The star schema is a symmetric structure. As such, the star schema is not biased toward facilitating a particular query.
- The symmetric star schema allows the designer to add new dimension tables with less disruption to the user's view of the data.

Snowflake Schema: A snowflake schema may be considered to be a compromise between a too-complex 3NF schema and a too-simple star schema. Compared to a 3NF schema, the snowflake schema does not allow arbitrarily complex relationships of any cardinality between any two tables. Compared to a star schema, the snowflake schema allows for the explicit representation of dimensional hierarchies.

5. Arguments against MD Schemas

We now assume a more conservative "DW=DB" philosophy and make observations that effectively place strong qualifications on the aforementioned advantages of MD schemas. Practitioners should consider the following caveats before committing to a MD schema.

5.1 Caveats Re. Correctness

Design correctness is the most important requirement. Therefore, designers are invited to review the advantages of a 3NF design and recognize that they are most relevant within a data warehouse environment.

Correct Semantics: A 3NF is "more correct" than a MD schema because it *directly* reflects the semantics of an application as represented within a semantic model. Furthermore, a 3NF schema is likely to be "more complete" because some designers might implement an overly simplified star schema, as illustrated in Figure 1.C, instead of a more complete, but more complex star schema, as illustrated in Figure 2.

Update Operations: Although a data warehouse is usually a read-only database, some on-line update operations may be executed. Zurek and Sinnwell make this point in [9] where they emphasize that the differences between real-world data warehouses and operational systems are not as "black and white" as some would believe. Therefore, the well-known integrity advantages of 3NF schemas apply to any relational database, including a data warehouse.

Schema Evolution: Whenever the real world changes, the semantic data model of the corresponding application domain may also change. This "Schema Evolution" problem is eternal and applies to practically all databases, including data warehouses. Because a 3NF schema, directly derived from a semantic model, is more stable than a non-3NF schema, schema evolution (and related ETL operations) will be more manageable with a 3NF schema.

5.2 Caveats Re. Efficiency

Machine efficiency pertains to physical database design. The designer should utilize internal DBMS facilities to support the important notion of *physical data independence*. This ability to improve performance by tuning internal access methods without making changes to the logical schema significantly reduces maintenance problems and costs. C. J. Date observes that advocates of star schemas appear to confuse logical design with physical design.

"*The problem is that there is really no concept of logical design, as distinct from physical design, in the star schema approach."* [1]

In principle, the DW=DB philosophy advocates acquiring sufficient hardware resources, a robust RDBMS, and then implementing an effective physical design. Recent advances in storage technology, parallel processing, data partitioning, physical access methods, materialized views, and query optimization make this possible for many data warehouse applications. For some applications, the additional cost to realize performance objectives with a 3NF schema may be less than the update processing and schema evolution costs associated with a MD schema.

The DW=DB philosophy recognizes the limitations of current technology. This philosophy also acknowledges that economic factors may prohibit acquisition of effective technological resources. Long before data warehousing became popular, practitioners modified OLTP logical schemas in order to enhance performance. Similar performance enhancing schema modifications apply within data warehouse systems. However, this fact does not necessarily imply that designers should explicitly target the construction of a MD schema. Instead, the designer should ask if it is possible to realize acceptable (not necessarily optimal) performance with a 3NF schema. *If and only if* this is not possible, then some effective design methodology should be utilized to produce an efficient *near-3NF* logical schema.

5.3 Caveats Re. Usability

The following questions should be considered before committing to a MD logical schema for reasons of usability.

1. Is a given 3NF schema really more complex than an equivalent MD schema? *If your real world is inherently complex, then your logical schema should represent this complexity, and your users must understand this complexity in order to accurately formulate their queries*. An overly simplified MD schema might increase, not reduce, usability problems. Spencer and Lewis observed that users of star schemas became confused about the semantics of a dimensional hierarchy that was stored within a single de-normalized dimension table. They concluded that the logical schema should be a snowflake. [7] Likewise, there may exist applications where a 3NF schema may indeed be perceived by users to be more understandable than a snowflake schema. Unfortunately, there appears to be little formal research that compares the usability of the three basic schemas. Therefore, we contend that practitioners should not necessarily reject a 3NF schema based on usability.

2. Is usability a database design consideration? The aforementioned usability advantages of MD schemas may be scientifically verifiable. However, these advantages might be realized by using effective front-end query and reporting tools. Therefore, given the availability of effective front-end tools, usability concerns alone would not justify transforming a 3NF schema into a MD schema. Unfortunately, the current consensus is that almost all tools have limitations. Hence, some applications will require designers to build friendly schemas.

3. Does usability apply to a *logical* schema? This is the *critical question*. Similar to Date's criticism that advocating a fast MD logical schema confuses logical design with physical design, we offer an additional parallel criticism. *Advocating a friendly logical schema confuses a logical schema with an external schema.* We suggest that designers consider using views to implement a *virtual star* on top of 3NF base tables. (Figure 4 illustrates an example.)

Figure 4: Views Support a Virtual Star

```
CREATE VIEW VSR  (S#, SNAME, SZIP,
                  R#, RNAME) AS
SELECT  S#, SNAME, SZIP, R.R#, RNAME
FROM    SUPPLIER S, REGION R
WHERE   S.R# = R.R#
```

6. Optimizer Considerations

A user might execute a query against view VSR that does not reference a column from REGION. The optimizer should (and some optimizers will) transform the query into an equivalent query that would not perform the unnecessary join with REGION. Unfortunately, not all optimizers are this smart. Designers should consider optimizer deficiencies when analyzing popular queries. If the performance penalty is significant, the designer may have to give up on the virtual star idea, or back away from the 3NF schema.

7. Conclusion

The DW=DB philosophy encourages practitioners to consider usability and performance when designing any database, including a data warehouse. The following guidelines apply to these issues.

Efficiency Guideline: Try to realize an acceptable level of performance by using powerful technology and effective physical design techniques. If performance requirements become so extreme, then some pragmatic modification of the logical schema could become necessary. As with OLTP systems, apply a *few* well-chosen logical design modifications to produce a *near*-3NF schema that directly reflects the semantics of the application and also satisfies performance objectives.

Usability Guideline: Some users may need to see a potentially complex 3NF schema because their application domain is inherently complex. Transforming a 3NF schema into a MD schema may actually obscure important semantic information necessary to formulate accurate queries. However, providing MD *external* schemas may be helpful for many users. Try to support MD schemas by using effective query and reporting tools. If such tools are not available, then use views to build virtual stars or snowflakes. Also, verify that your optimizer is smart enough to avoid the kind of problem described in Section 6.

The DW=DB philosophy requires that designers initially formulate a 3NF schema and then "back away" from the 3NF schema if and only if the payoff is significant. Making special case design modifications to a 3NF schema is much less radical than adopting a specialized design methodology that specifically targets a MD schema. Therefore, the DW=DB philosophy is more compatible with a methodology that builds a 3NF schema and subsequently generates a near-3NF schema, or even a MD schema. The development of a 3NF schema, even as an intermediate result, ensures that the design is built on a solid foundation. Furthermore, staying within the context of traditional database design will eliminate potential organizational problems, and training costs, associated with adopting a novel or specialized design methodology.

References

[1] Date, C. J., An Introduction to Database Systems (7th), Addison-Wesley, 2000.

[2] Golfarelli, M., and Rizzi, S., "A Methodological Framework for Data Warehouse Design," DOLAP, 1998.

[3] Hellerstein,, Stonebraker, and Caccia, "Independent, Open Enterprise Data Integration," Bulletin of IEEE Committee on Data Engineering, 1999.

[4] Jarke, M., Lenzerini, M., and Vassiliou, Y, Fundamentals of Data Warehousing (2nd ed.), Springer-Verlag, 2002.

[5] Kimball, R., The Data Warehouse Toolkit, Wiley, 1996.

[6] Kimball and Ross, The Data Warehouse Toolkit (2nd ed.), Wiley, 2002.

[7] Spencer, T. and Loukas, T., "From Star to Snowflake to ERD: Comparing Data Warehouse Design Approaches," Enterprise Systems Journal, 10/99.

[8] Vassiliadis P, and Sellis, T. "A Survey of Logical Models for OLAP Databases," SIGMOD Record, 12/99.

[9] Zurek, T., and Sinnwell, M., "Data Warehousing Has More Colours Than Just Black & White," Proc. 25th VLDB Conference, 1999.

[10] A longer version of this paper, with a short case study, can be obtained at www.rh.edu/~martyn/warehouse.